FIWARE ZONE

Práctica IoT

https://www.fiware.org
https://lab.fiware.org
Follow @FIWARE on Twitter!

https://fiware.zone
Follow @FIWAREZone on Twitter!

Telefónica

JUNTA DE ANDALUCIA
CONSEJERÍA DE EMPLEO, EMPRESA Y COMERCIO

Conocimientos previos

Se emplearán las siguientes tecnologías, por lo que es conveniente tener un conocimiento previo de:

- **C++:** Para la programación del microcontrolador (Arduino)
- **HTML:** Para la página web y los formularios que crea el programa
- **API REST:** Para gestionar la API de FIWARE
- **FIWARE – IoT Agent Ultraligth 2.0:** Para el envío de datos por parte del dispositivo

01

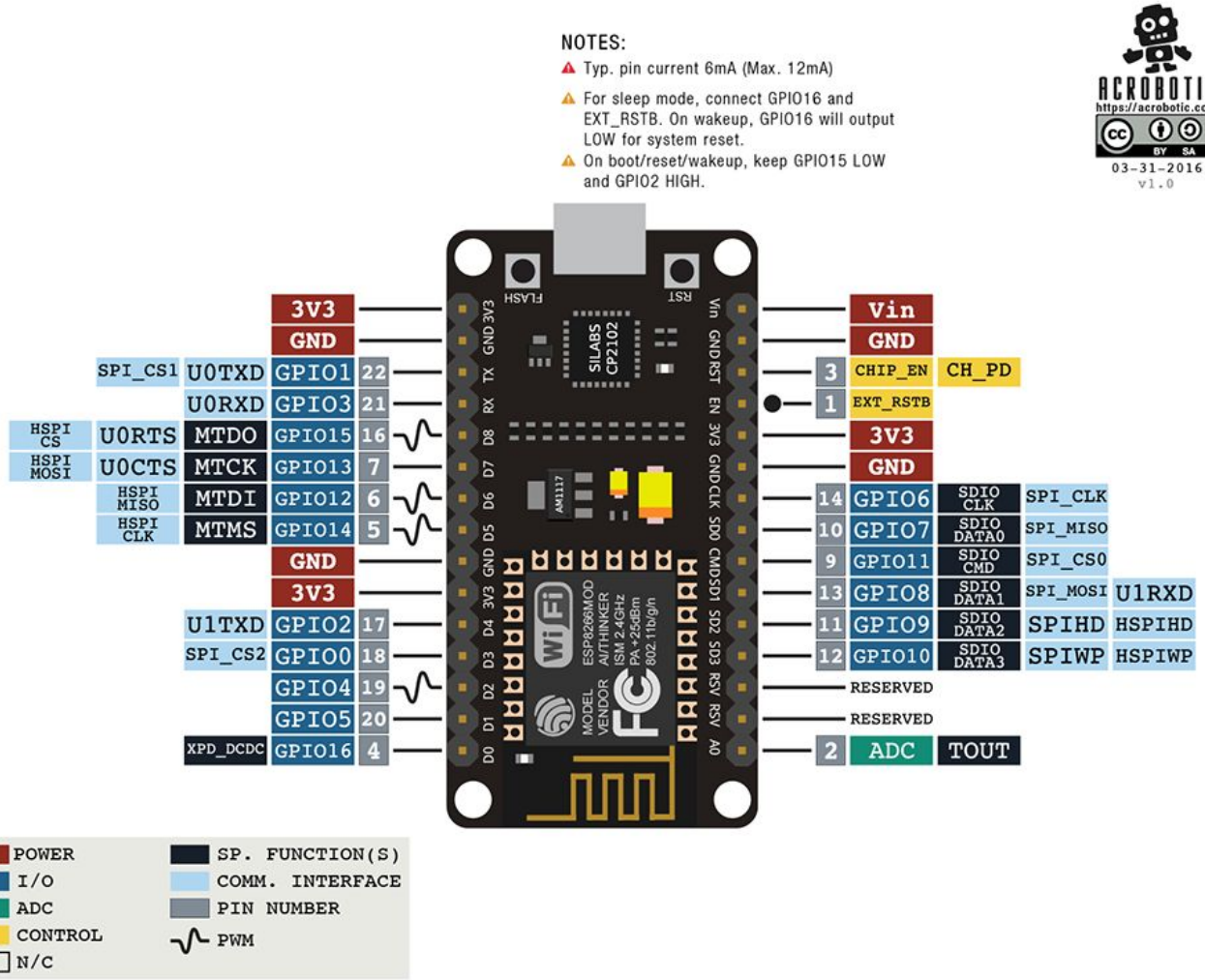
Descripción del Hardware

Placa de desarrollo

Specs

ESP8266
RAM: 128KB
ROM: 4MB
Alimentación: 3.3V
WiFi integrado
13 Pin IO
USB<->UART incluido
Regulador 5V (USB) a 3.3V

Pinout



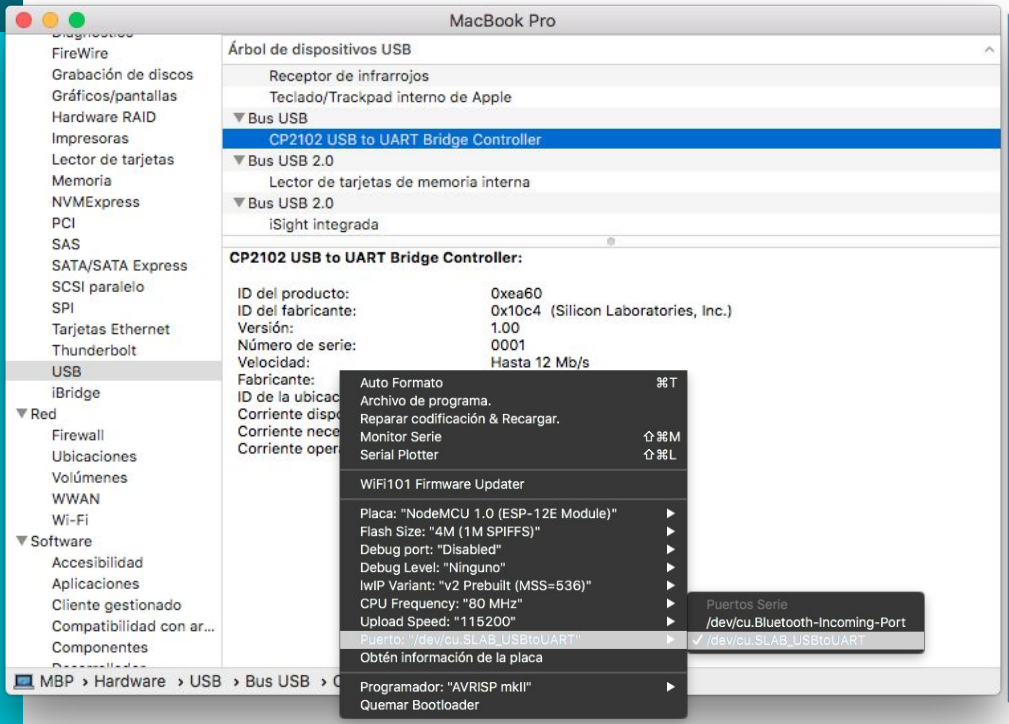
FIWARE ZONE

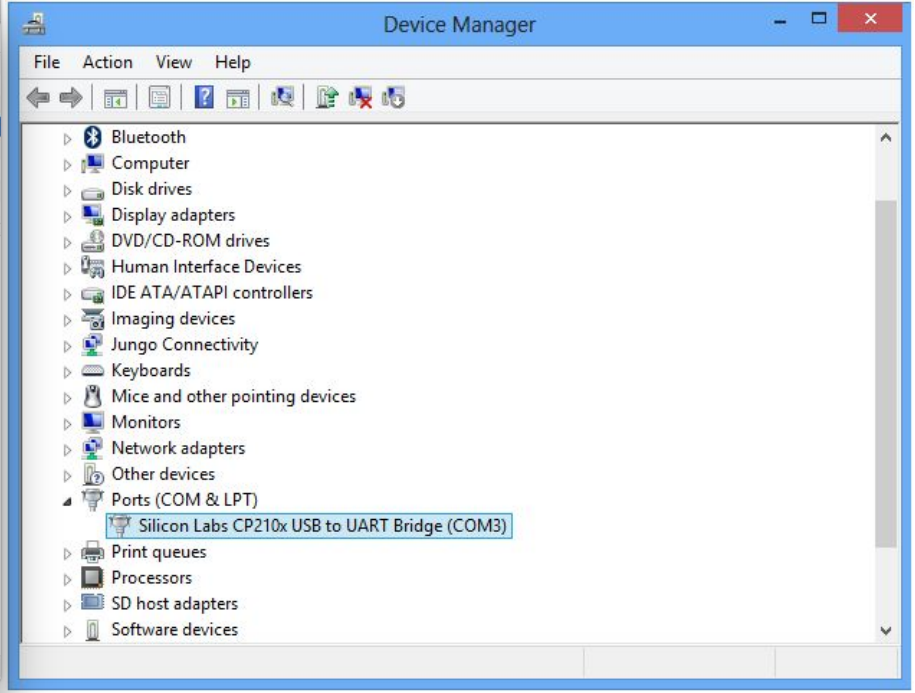
Telefónica

JUNTA DE ANDALUCIA

02 Instalación del IDE

2.1 Drivers

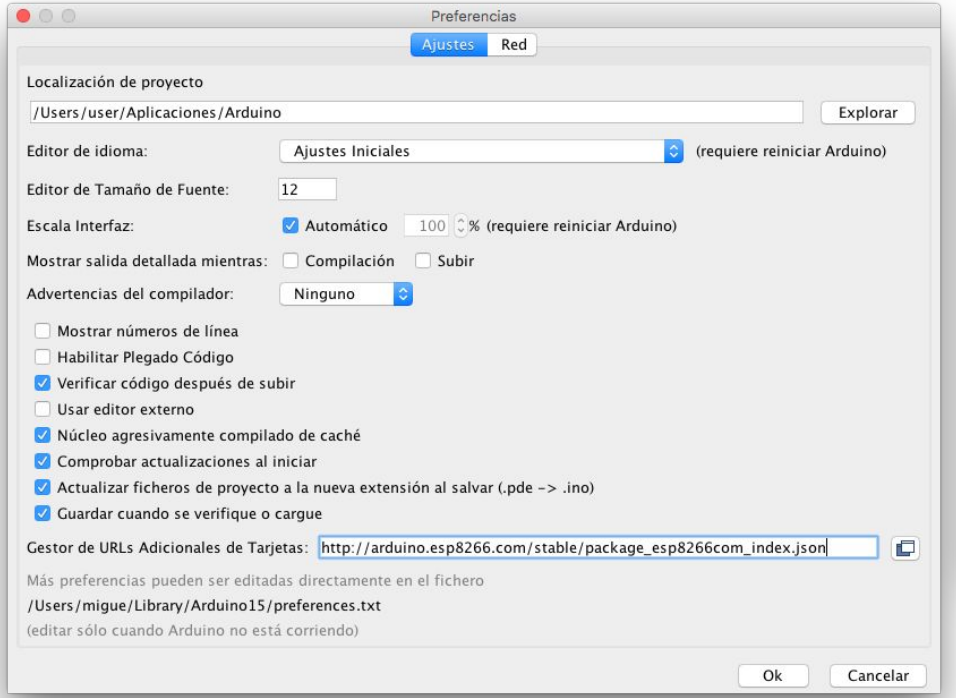




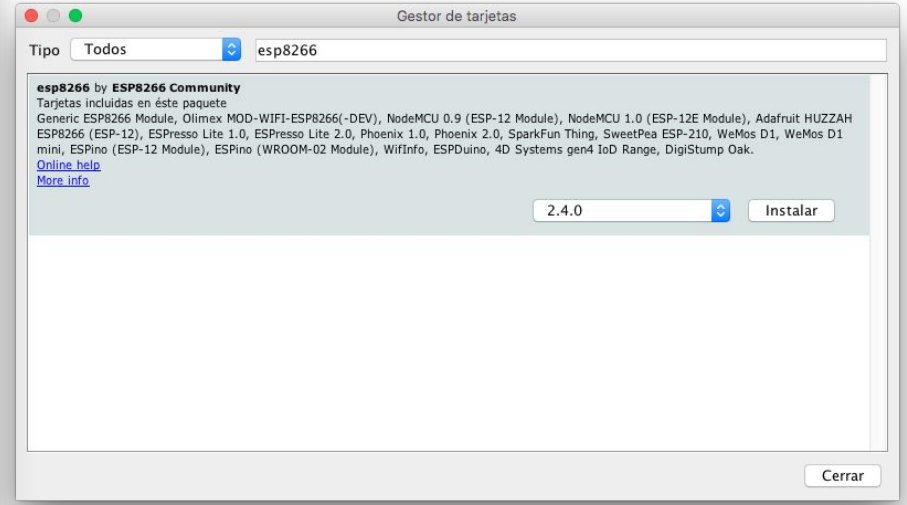
www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

2.2 Arduino

1

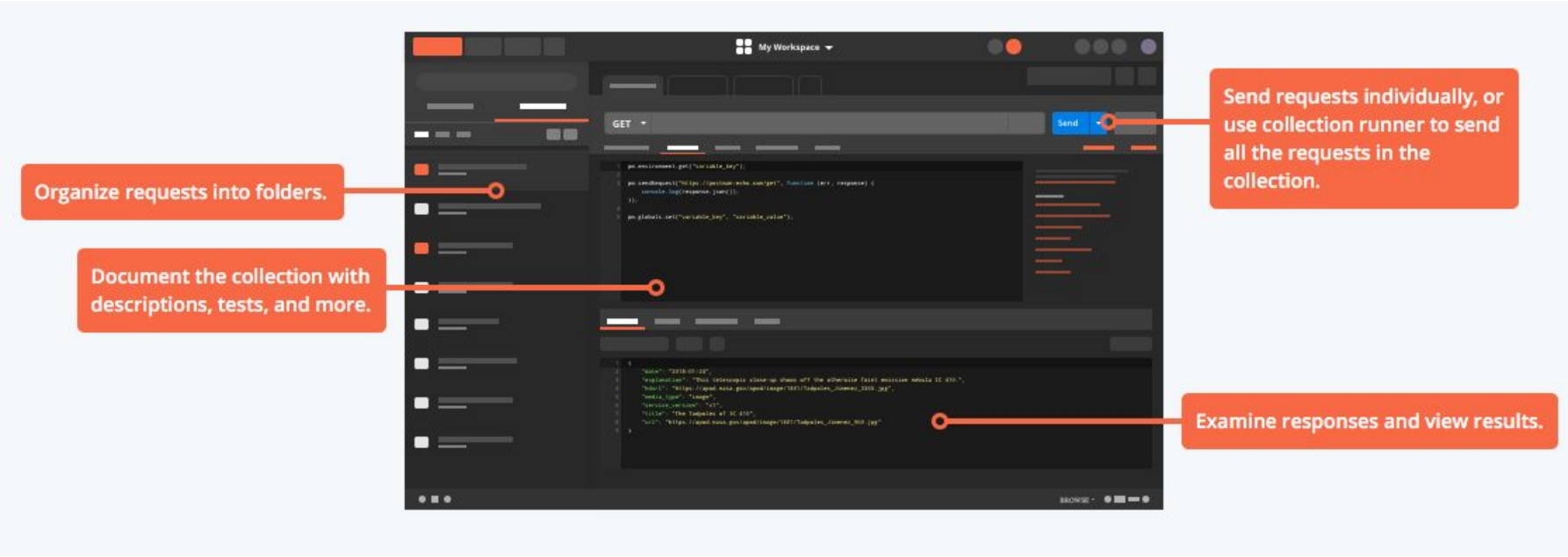


2



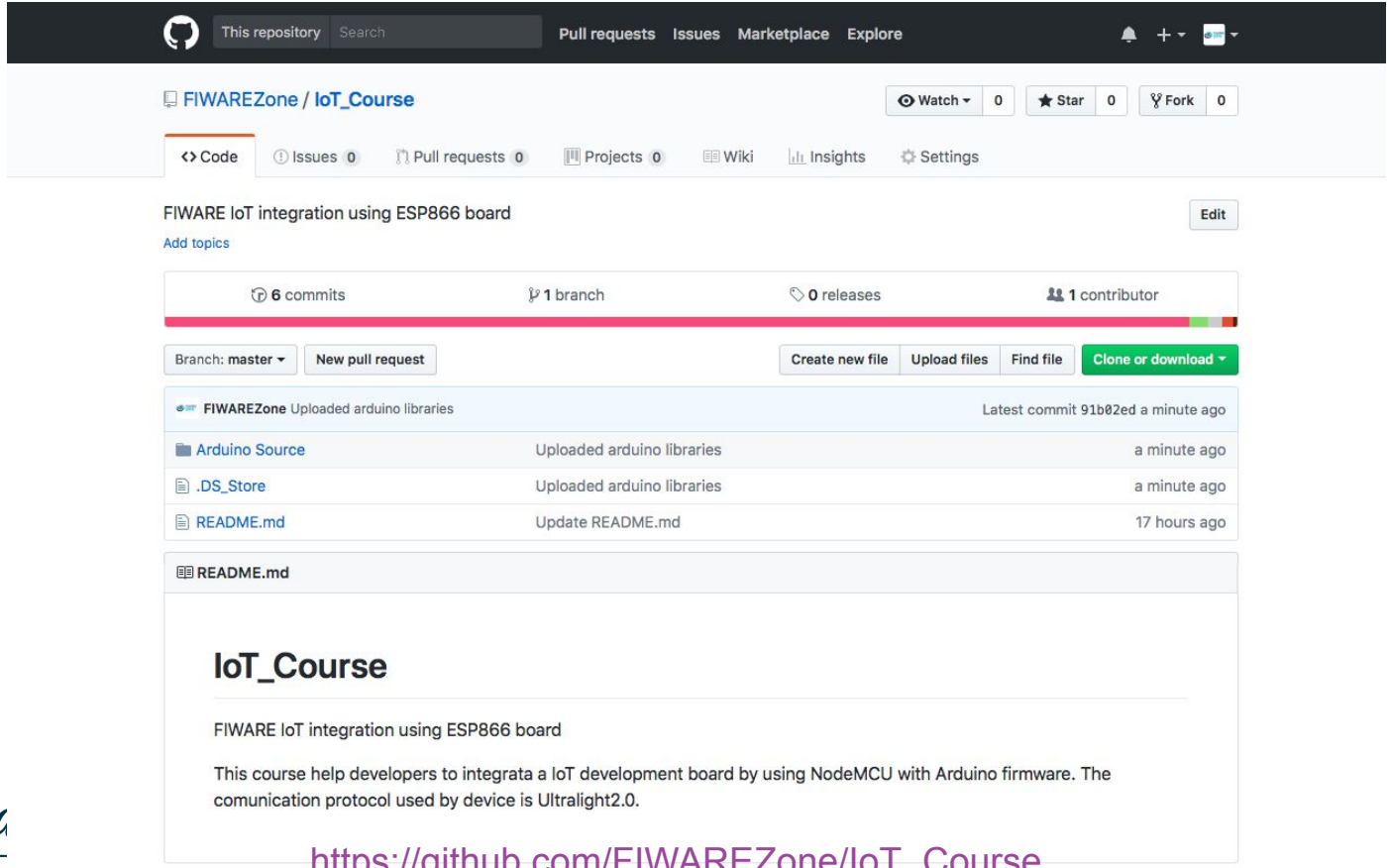
http://arduino.esp8266.com/stable/package_esp8266com_index.json

2.3 Postman



https://www.getpostman.com

2.3 Repositorio



https://github.com/FIWAREZone/IoT_Course

03

Provisionamiento de dispositivo

3.1 Creando la entidad

HTTP POST:

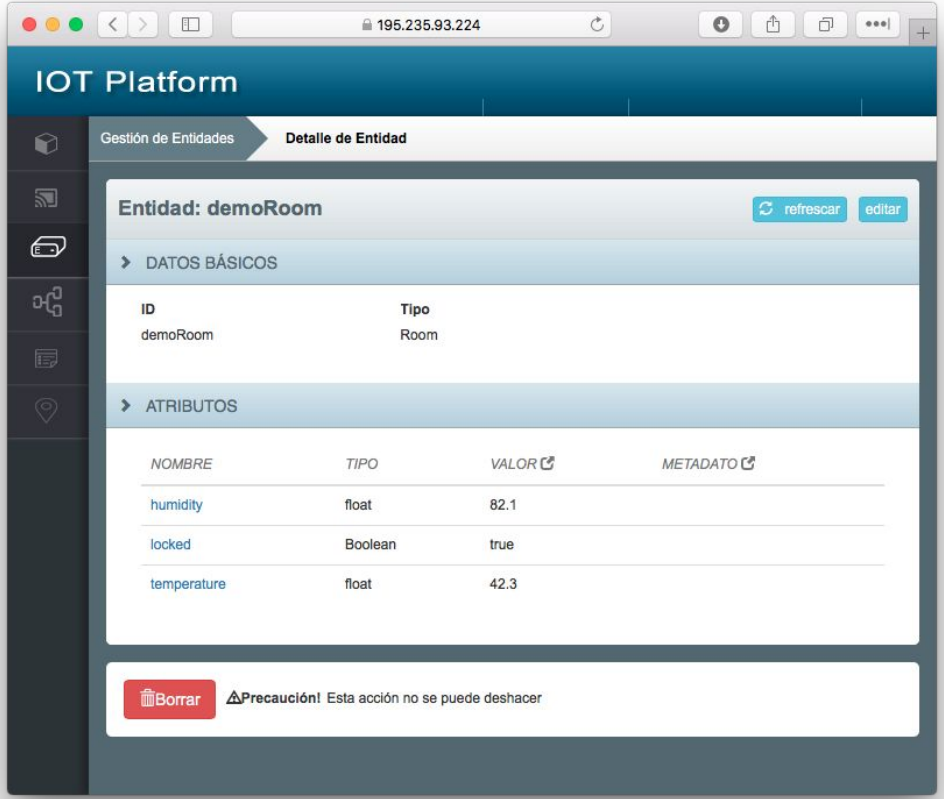
https://{{host}}:10027/v2/entities

Headers: {'content-type': 'application/json'; "Fiware-Service: **{{service}}**"; "Fiware-ServicePath:**{{subservice}}**"; "X-Auth-Token: **{{token}}**"}

Payload:

```
{
        "id": "demoRoom",
        "type": "Room",
        "locked": {
                "value": true,
                "type": "Boolean"
        },
        "temperature": {
                "value": 42.3,
                "type": "float"
        },
        "humidity": {
                "value": 82.1,
                "type": "float"
        }

}
```

3.2 Creando la entidad



3.2 Creando el dispositivo

HTTP POST:

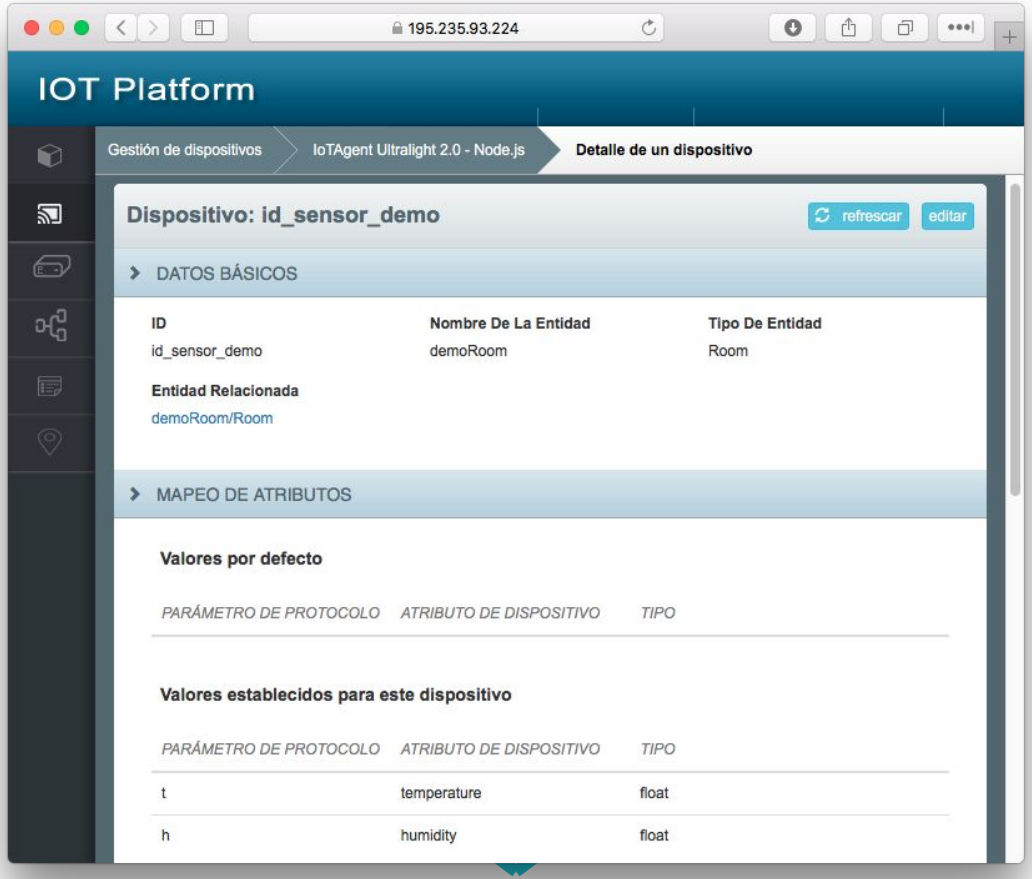
https://{{host_iota}}:8088/iot/devices

Headers: {'content-type': 'application/json'; "Fiware-Service: **{{service}}**"; "Fiware-ServicePath:**{{subservice}}**"; "X-Auth-Token: **{{token}}**"}

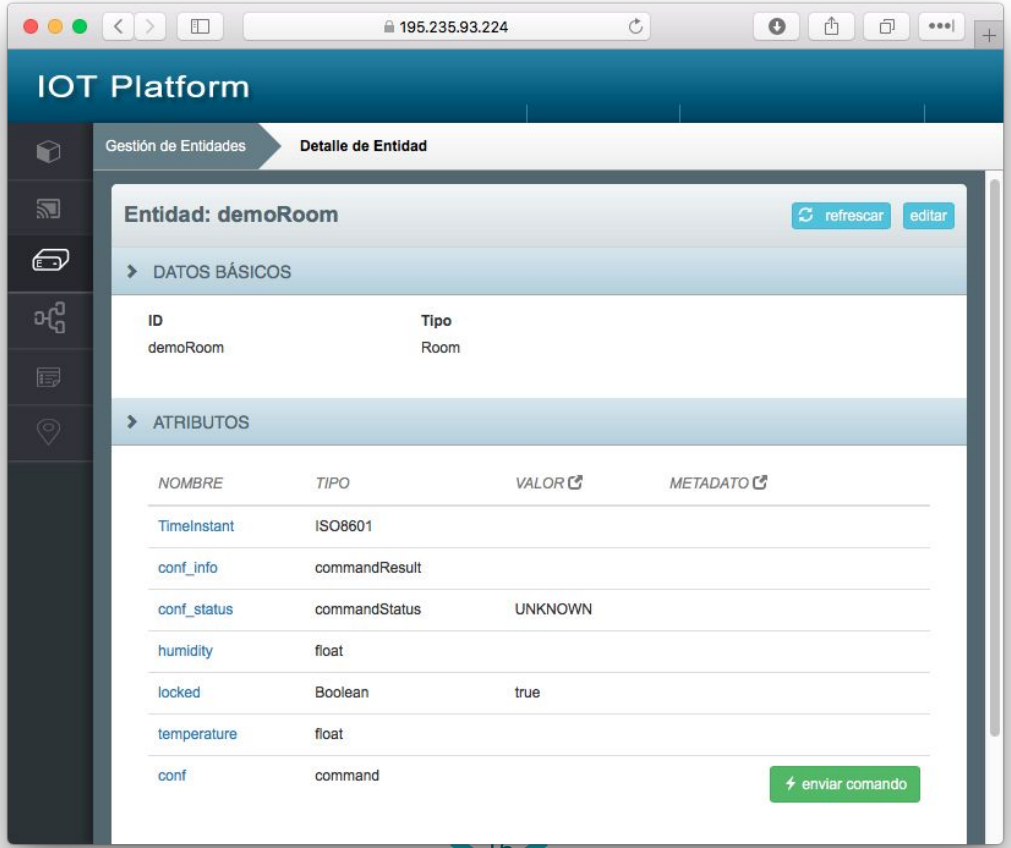
Payload:

```
{
   "devices": [
      {
         "device_id": "id_sensor_demo",
         "entity_name": "demoRoom",
         "entity_type": "Room",
         "attributes": [
             { "object_id": "t", "name": "temperature", "type": "float" },
             { "object_id": "h", "name": "humidity", "type": "float" }
         ],
         "lazy":[ ],
         "commands": [
             { "object_id": "conf", "name": "conf", "type": "command" }
         ],
         "static_attributes": [ ],
     "protocol": "IoTA-UL",
     "transport": "HTTP"
      }
   ]
}
```

3.2 Creando el dispositivo



3.2 Creando el dispositivo. Entidad



3.3 Enviando datos por ultralight

HTTP POST:

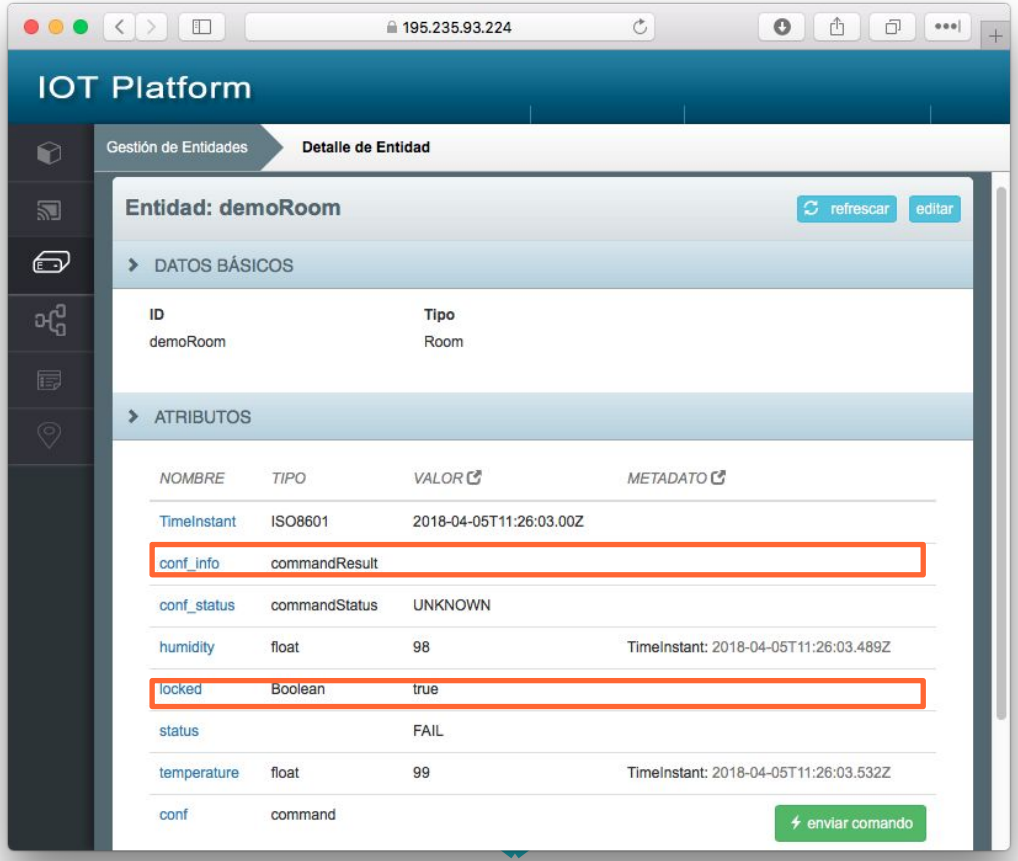
http://{{host_iota}}:8085/iot/d?k={{UL_apikey}}&i={{device_ID}}

Headers: {'content-type': text/plain’}

Payload:

t|99#h|98

3.3 Enviando datos por ultralight



3.4 Enviando datos al dispositivo

HTTP POST:

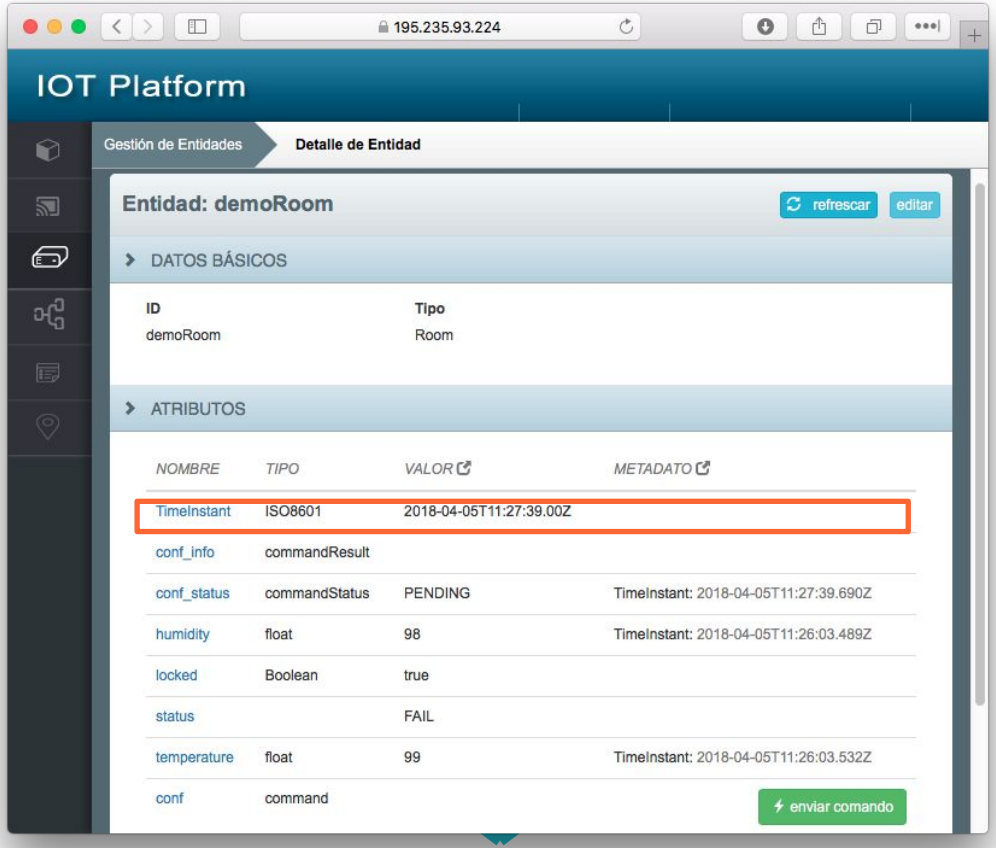
https://{{host}}:10027/v1/contextEntities/type/{{device_type}}/id/{{entity_name}}/attributes/conf

Headers: {'content-type': 'application/json'; "Fiware-Service: **{{service}}**"; "Fiware-ServicePath:**{{subservice}}**"; "X-Auth-Token: **{{token}}**"}

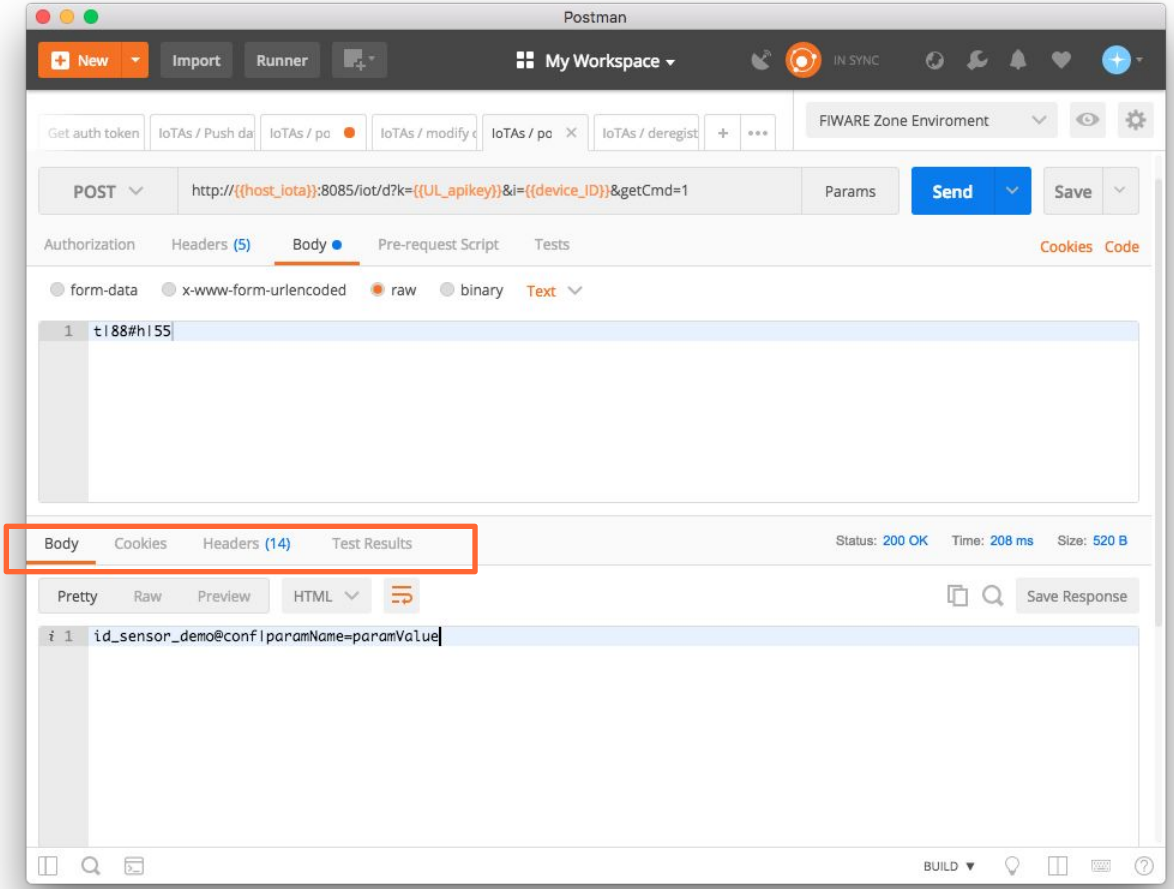
Payload:

{"value":{"paramName":"paramValue"},"type":"command"}

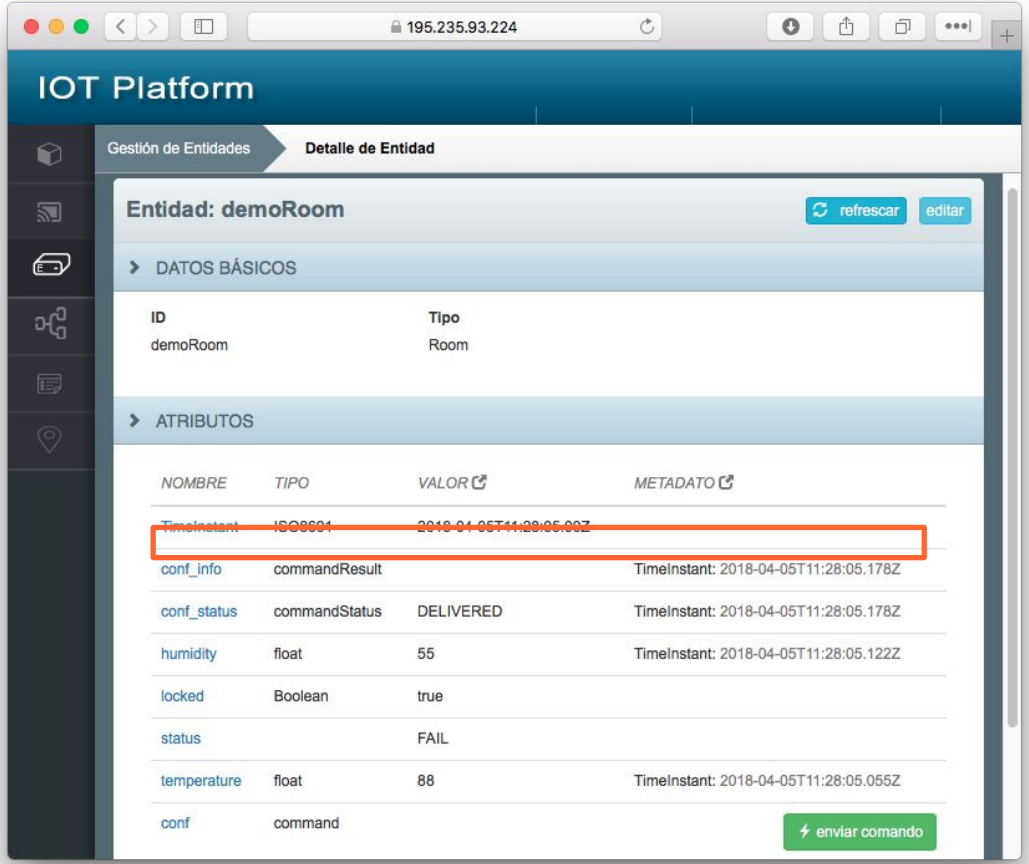
3.4 Enviando datos al dispositivo



3.4 Enviando datos al dispositivo



3.4 Enviando datos al dispositivo



04

Programa y funcionamiento

4.1 Variables de configuración

```
/*************** Global Variables ***************/
//FIWARE Variables
String FIWARE_device_ID = "SENSOR_ID";
String FIWARE_server = "195.235.93.235";
String FIWARE_port = "8085";
String FIWARE_apikey = "REEMPLAZAR_APIKEY";

//Wifi Variables
const char* WiFi_Network = "NOMBRE_WIFI";
const char* WiFi_Password = "PASS_WIFI";

const char* WiFi_SoftAP_Name = "FIWAREZone_IoT";
const char* WiFi_SoftAP_WiFi_Name = "FIWAREZone_IoT_Wifi";
```

4.2 Librerías empleadas

```
#include <FS.h>                   //this needs to be first, or it all crashes and burns

//ESP8266 Native libraries
#include <ESP8266WiFi.h>          //https://github.com/esp8266/Arduino
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <ESP8266HTTPUpdateServer.h>
#include <ESP8266HTTPClient.h>

//External libraries
#include <WiFiManager.h>          //https://github.com/tzapu/WiFiManager
#include <ArduinoJson.h>          //https://github.com/bblanchon/ArduinoJson
```

ESP8266 Arduino Core

latest

Search docs

CONTENTS:

- Installing
- Reference
- Libraries
- File system
- ESP8266WiFi
- OTA Updates
- PROGMEM
- Boards
- FAQ
- Exception causes
- Debugging
- Stack Dump
- Using with Eclipse

Read the Docs v: latest

4.2.1 Librerías Nativas ESP8266

- ESP8266AVRISP
- ESP8266HTTPClient
- ESP8266HTTPUpdateServer
- ESP8266LLMNR
- ESP8266NetBIOS
- ESP8266SSDP
- ESP8266WebServer
- ESP8266WiFi
- ESP8266WiFiMesh
- ESP8266httpUpdate
- ESP8266mDNS

https://github.com/esp8266/Arduino

4.2.2 Librerías externas

WiFiManager

Permite configurar un punto de acceso. Si no lo encuentra, crea red propia para configurar el AP conectándote mediante wifi. Guarda las credenciales en la memoria interna

https://github.com/tzapu/WiFiManager

4.2.2 Librerías externas

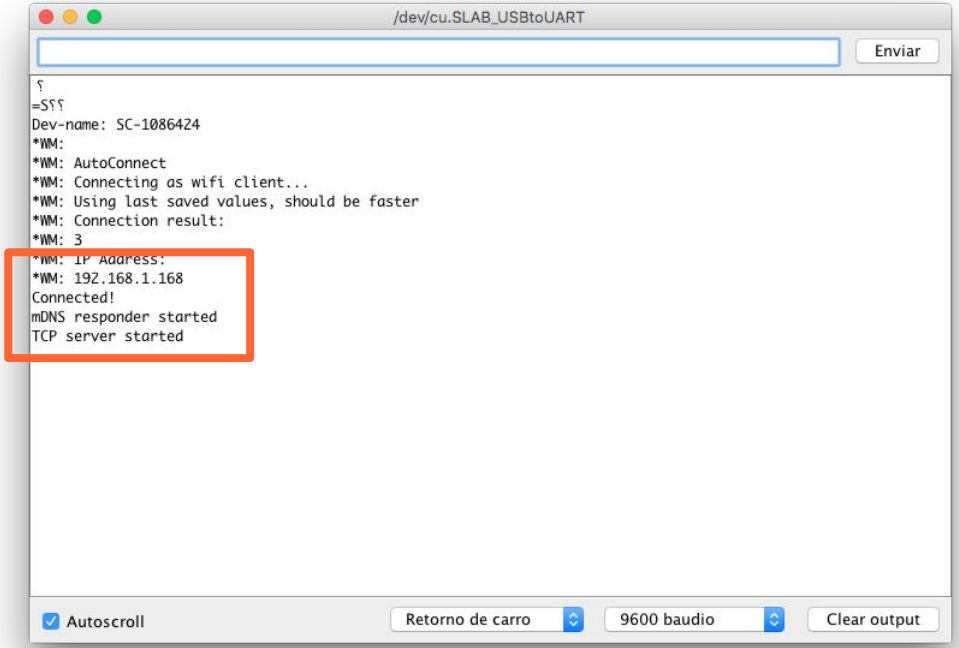
ArduinoJson

Serialización y deserialización de JSON de forma sencilla para el envío de datos

https://github.com/bblanchon/ArduinoJson

4.3 Configuración WiFi

Revisamos el monitor del puerto serie
Ya está conectado a la Wifi que hemos configurado. Ahora podemos acceder desde el navegado, conectado a la misma red



4.4 Servidor Web.Página principal

FIWARE.ino
void setup()

```
//Bind Webserver URL functions
server.on ( "/", handleRoot );
server.on ( "/wrst", handleWrst );
server.on ( "/wifi", handleWifi );
server.on ( "/webota", handleWebota );
server.on ( "/help", handleHelp );
server.on ( "/postul2", handlePostUL2 );
server.on ( "/postul2data", handlePostUL2data );
```

router.ino
Todas las funciones que renderizan las páginas

```
void handleRoot() {

  String page = FPSTR(HTTP_HEAD);
  page.replace("{v}", "FIWARE_Zone_IoT");
  page += FPSTR(HTTP_STYLE);
  page += FPSTR(HTTP_HEAD_END);
  page += F("<h1>FIWARE IoT Device</h1>");
  page += F("<h3>Menu</h3>");
  page += FPSTR(HTTP_MAIN_FORM);
  page += String("hostname:");
  page += String(dev_hostname);
  page += String(".local");
  page += FPSTR(HTTP_END);
  server.send(200, "text/html", page);

}
```

4.4 Servidor Web. Página principal

router.ino

const char HTTP_MAIN_FORM[ ]

```
const char HTTP_MAIN_FORM[] PROGMEM  = "<form action=\"/wifi\" method=\"get\"><button>Setup Wifi</button></form><br/><form action=\"/wrst\" method=\"g
```

```
const char HTTP_MAIN_FORM[] PROGMEM  =
<form action="/wifi" method="get"><button>Setup Wifi</button></form><br/>
<form action="/wrst" method="get"><button>Clear wifi setup</button></form><br/>
<form action="/webota" method="get"><button>Update</button></form><br/>
<form action="/postul2" method="get"><button>POST UL2</button></form><br/>
<form action="/help" method="get"><button>Help</button></form><br/>
```

Utilidades de interés:

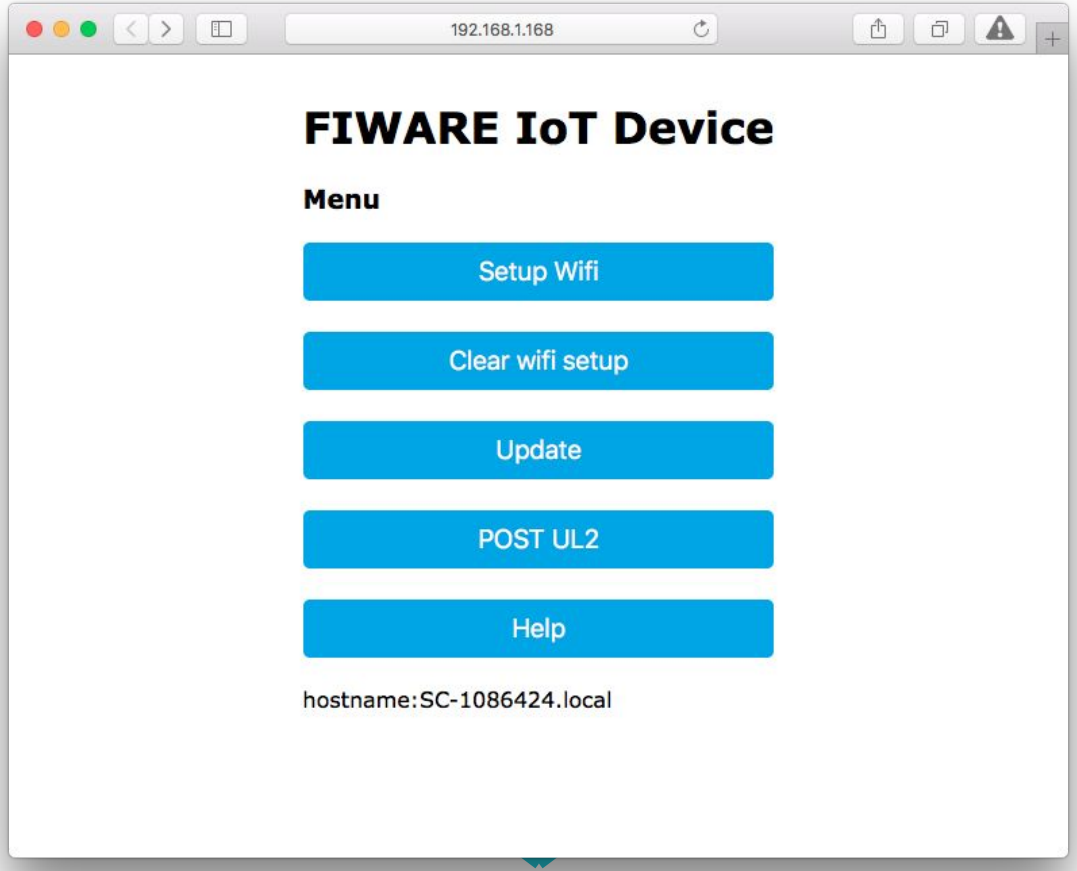
Convertidor string C (caracteres de escape)

http://tomeko.net/online_tools/cpp_text_escape.php?lang=en

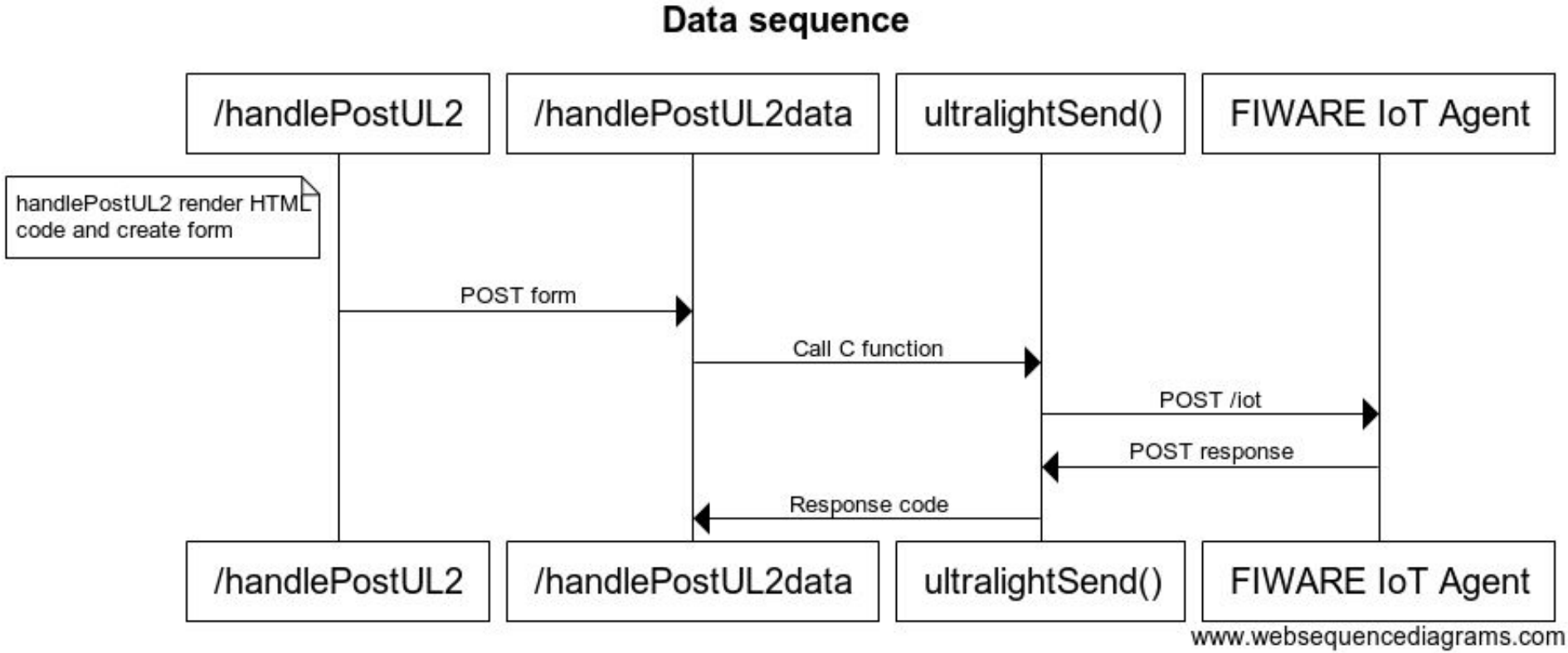
Formateador de HTML

http://minifycode.com/html-beautifier/

4.4 Servidor Web.Página principal

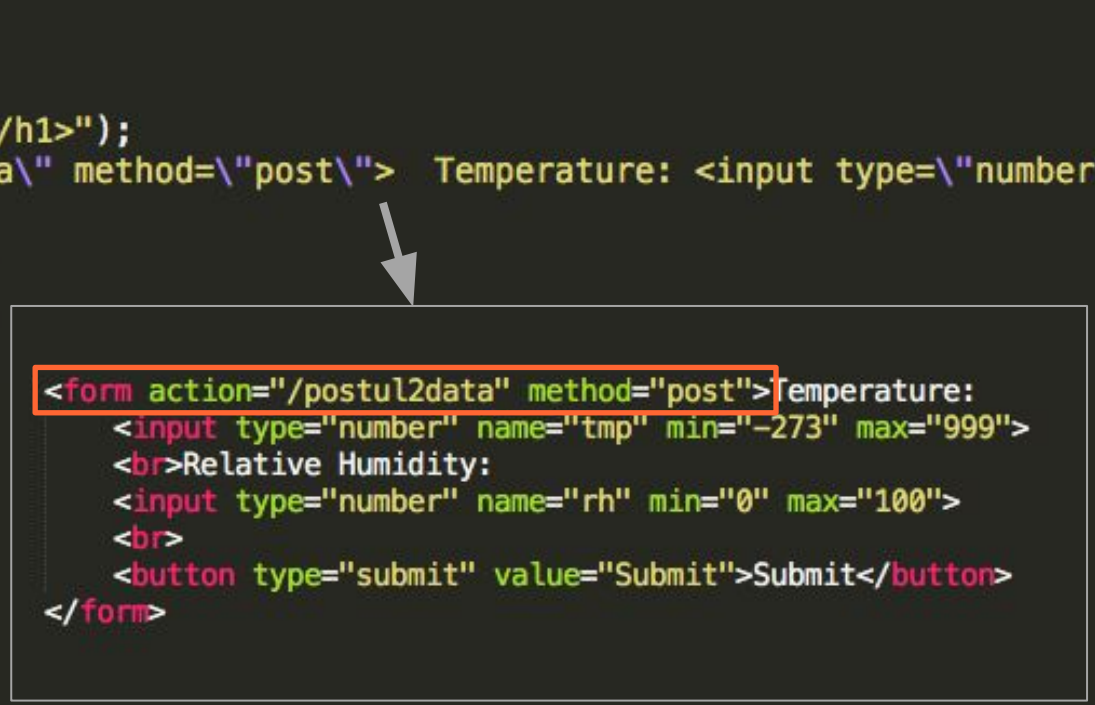


4.5 Secuencia de envío

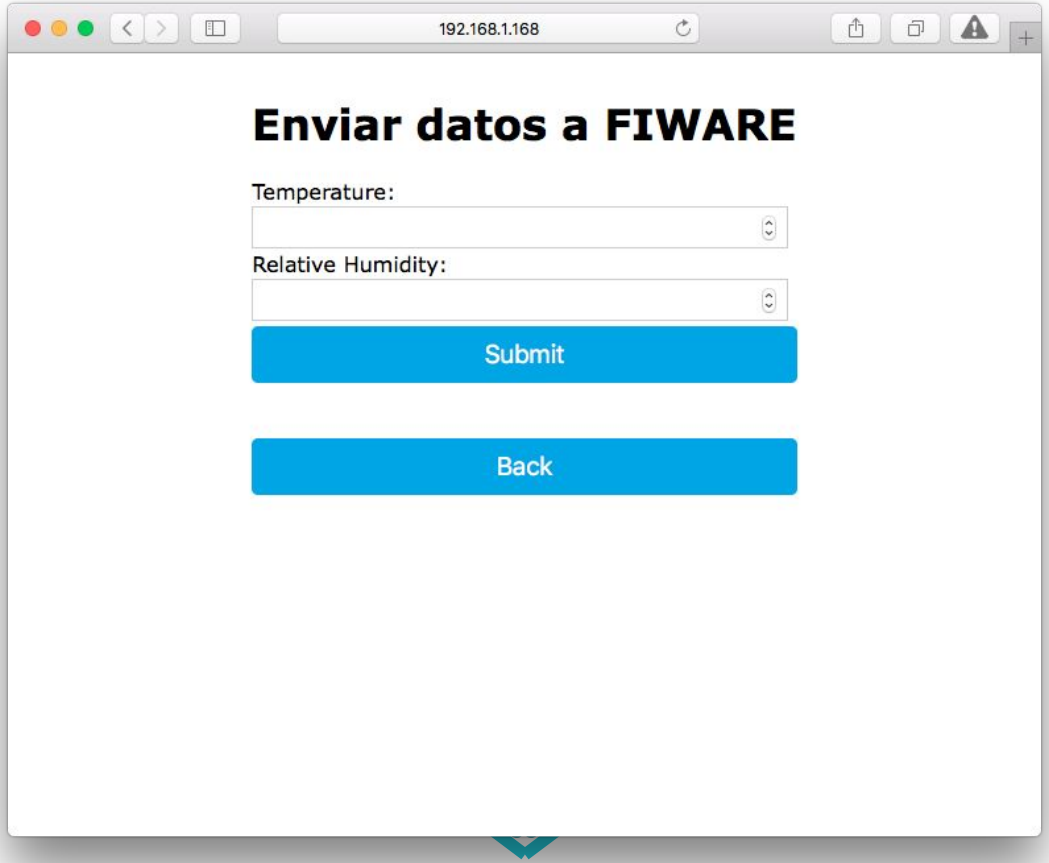


4.6.1 Función de envío a FIWARE

```
void handlePostUL2(){
  String page = FPSTR(HTTP_HEAD);
  page.replace("{v}", "Info");
  page += FPSTR(HTTP_SCRIPT);
  page += FPSTR(HTTP_STYLE);
  page += FPSTR(HTTP_HEAD_END);
  page += F("<h1>Enviar datos a FIWARE</h1>");
  page += F("<form action=\"/postul2data\" method=\"post\">  Temperature: <input type=\"number"
  page += FPSTR(HTTP_BACK_BUTTON);
  page += FPSTR(HTTP_END);
  server.send(200, "text/html", page);
}
```



4.6.2 Panel de envío a FIWARE



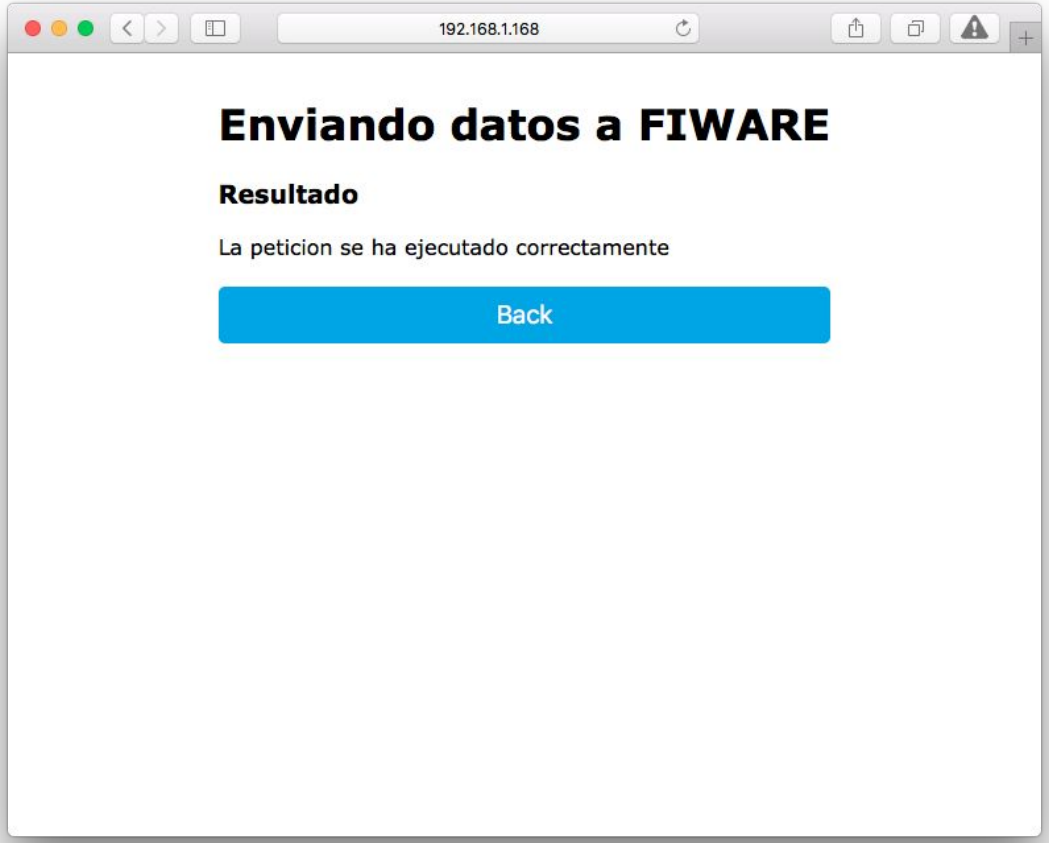
4.6.3 Llamada a función de envío

```
void handlePostUL2data(){

    //Search form post values
    String tmp, rh;
      if (server.args() > 0 ) {
          for ( uint8_t i = 0; i < server.args(); i++ ) {
            if (server.argName(i) == "tmp") {
               // do something here with value from server.arg(i);
               tmp = server.arg(i);
            }
            else if(server.argName(i) == "rh"){
              rh = server.arg(i);
            }
        }
      }


    //POST Data
    String page = FPSTR(HTTP_HEAD);
    page.replace("{v}", "Info");
    page += FPSTR(HTTP_SCRIPT);
    page += FPSTR(HTTP_STYLE);
    page += FPSTR(HTTP_HEAD_END);
    page += F("<h1>Enviando datos a FIWARE</h1>");

    //Parse response
    int returnCode = ultralightSend(FIWARE_server,FIWARE_port,FIWARE_token,FIWARE_ID,"temp|"+tmp+"#hr|"+rh);
    if (returnCode ==200){
        page += F("<h3>Resultado</h3>");
        page += F("La peticion se ha ejecutado correctamente");
    }
    else{
        page += F("<h3>Resultado</h3>");
        page += F("Ha habido un error en la peticion");
        page += String(returnCode);
    }

    //End page
    page += FPSTR(HTTP_BACK_BUTTON);
    page += FPSTR(HTTP_END);
    server.send(200, "text/html", page);

}
```

4.6.4 Función de envío por Ultralight 2.0

```
int ultralightSend (String URL, String port, String Token, String ID, String Body) {
  int httpCode = 0;

  Serial.println("http://" + URL + ":" + port + "/iot/d?k=" + Token + "&i=" + ID + "&getCmd=1");
  http.begin("http://" + URL + ":" + port + "/iot/d?k=" + Token + "&i=" + ID + "&getCmd=1");          //Specify request destination

  http.addHeader("Content-Type", "text/plain");         //Specify content-type header

  httpCode = http.POST(Body);                           //Send the Body
  String payload = http.getString();                    //Get the response payload

  Serial.println("Request Done");
  Serial.print("Return code: ");
  Serial.println(httpCode);                             //Print HTTP return code
  Serial.print("Response payload: ");
  Serial.println(payload);                              //Print request response payload

  http.end();                                           //Close connection
  return httpCode;
}
```

4.6.5 Respuesta de envío



05 Otras funcionalidades

5.1 WiFi Manager

```
#define USE_CREDENTIALS
```

Comentar

```
/************** Global Variables **************/
//FIWARE Variables
String FIWARE_device_ID = "SENSOR_ID";
String FIWARE_server = "195.235.93.235";
String FIWARE_port = "8085";
String FIWARE_apikey = "REEMPLAZAR_APIKEY";

//Wifi Variables
const char* WiFi_Network = "NOMBRE_WIFI";
const char* WiFi_Password = "PASS_WIFI";

const char* WiFi_SoftAP_Name = "FIWAREZone_IoT";
const char* WiFi_SoftAP_WiFi_Name = "FIWAREZone_IoT_Wifi";
```

Modificar

5.1 WiFi Manager

Primer inicio

La primera vez que arranca el dispositivo, al no tener una red configurada, este crea una red propia para configurar la red wifi a la que tiene que conectarse la placa. La red que crea tendrá el nombre de la variable definida como **WiFi_SoftAP_Name**.

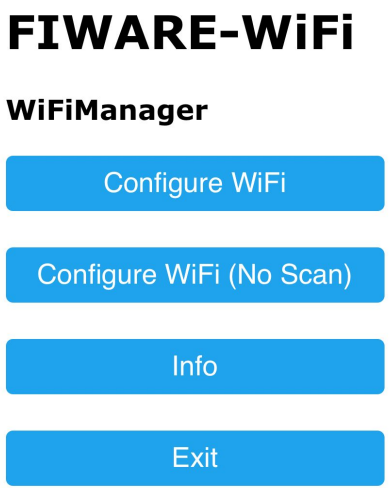
La IP a la que hay que ir una vez conectados es :**192.168.4.1**

5.1 WiFi Manager

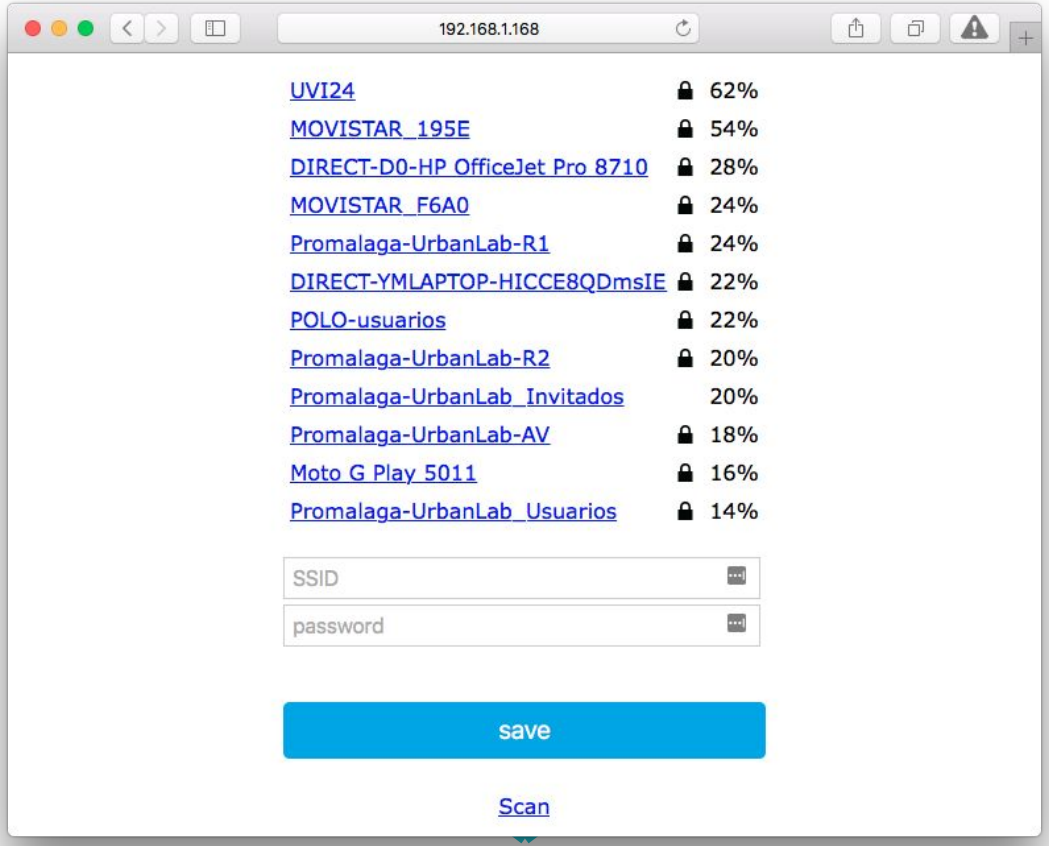
Configurar WiFi

Al entrar en Setup Wifi, el dispositivo se reiniciará en modo configuración de wifi creando una nueva red con el nombre de **WiFi_SoftAP_WiFi_Name**. Al entrar a esta nueva red, nos saldrá un portal cautivo en el que podemos configurar la red wifi.

En caso de que no salga dicho portal, nos conectamos a la misma IP de antes: **192.168.4.1**



5.1 WiFi Manager

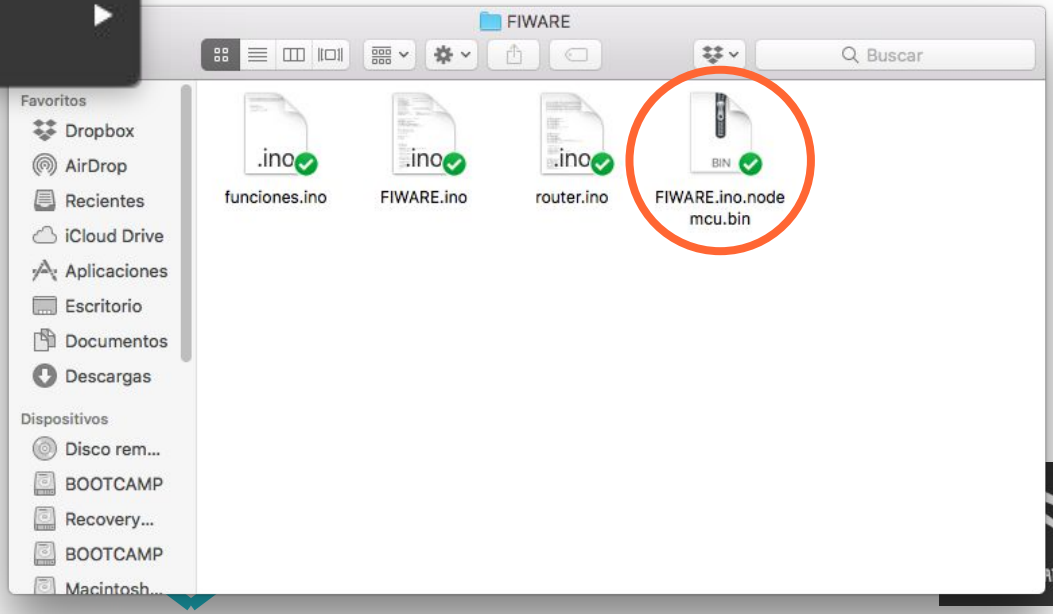


5.2 OTA

```
//OTA
const char* update_path = "/webota";
const char* update_username = "admin";
const char* update_password = "admin";
```

5.2 OTA


Arduino
Archivo
Editar
Programa
Herramientas
Ayuda
Verificar/Compilar ⌘R
Subir ⌘U
Subir Usando Programador ⇧⌘U
Exportar Binarios compilados ⌥⌘S
Mostrar Carpeta de Programa ⌘K
Incluir Librería
Añadir fichero...


FIWARE
Buscar
Favoritos
Dropbox
AirDrop
Recientes
iCloud Drive
Aplicaciones
Escritorio
Documentos
Descargas
Dispositivos
Disco rem...
BOOTCAMP
Recovery...
BOOTCAMP
funciones.ino
FIWARE.ino
router.ino
FIWARE.ino.node mcu.bin

5.3 Datalogger

Proyecto Datalogger.ino

Ejemplo que envía los datos periódicos de los datos de un sensor.

Lectura de un sensor por **I2C** modelo **BMP085** con las siguientes medidas:

- Presión
- Temperatura
- Altitud

5.3 Datalogger

```
// I2C Barometric sensor libraries
#include <Wire.h>
#include <Adafruit_BMP085.h>
#define SCL_PIN D1
#define SDA_PIN D2
Adafruit_BMP085 bmp;

unsigned long previousMillis=0;
unsigned long deltaMillis=15000;
void dataSenderCallback();
```

Tiempo de envío

```
// I2C Sensor Initialization
Wire.pins(SDA_PIN, SCL_PIN);
Wire.begin(SDA_PIN, SCL_PIN);
if (!bmp.begin()) {
    Serial.println("No BMP180 / BMP085");// we dont wait for this
    while (1) {}
}
```

5.3 Datalogger

```
void loop() {
  //WebServer task
  server.handleClient();


  if (millis()>previousMillis+deltaMillis){
    previousMillis=millis();
    //Execute task
    dataSenderCallback();
  }
}

String dataReaderSensor(){
    String t = "T=" + String(bmp.readTemperature()) + " *C";
    String p = "P=" + String(bmp.readPressure()) + " Pa";
    String a = "A=" + String(bmp.readAltitude(101325)) + " m";// insert pressure at sea level

  Serial.println("Reading data from sensors:");
  Serial.println(t);
  Serial.println(p);
  Serial.println(a);

  String r = "t|"+ String(bmp.readTemperature()) +"#p|"+ String(bmp.readPressure()) +"#a|"+ String(bmp.readAltitude(101325));
  return r;
}

void dataSenderCallback(){
  int returnCode = ultralightSend(FIWARE_server,FIWARE_port,FIWARE_apikey,FIWARE_device_ID,dataReaderSensor());
   if (returnCode ==200){
     Serial.println("Data sent successfully to FIWARE");
    }
    else{
     Serial.println("Failure sending data to FIWARE");
    }
}
```

@FIWAREZone

fiware.zone@fiware.zone